Definiton project

**1. RoomRepository Class**

**Purpose**:  
This class is responsible for managing the database interactions related to room data in the hotel management system. It implements the IRoomRepository interface to provide methods for creating, retrieving, updating, and deleting room information.

**Key Methods**:

* **createRoom**: Inserts a new room record into the database.
* **getRoomById**: Retrieves room details by its unique identifier.
* **getAllRooms**: Fetches all the rooms available in the database.
* **deleteAllRooms**: Deletes all room records from the database and resets the room ID sequence.
* **getRoomTypes**: Retrieves all unique room types available in the system.
* **getAvailableRoomsByType**: Finds available rooms of a specific type that are not currently booked.
* **resetRoomIdSequence**: Resets the sequence for room IDs, used to ensure correct auto-increment behavior.

**2. HotelApplication Class**

**Purpose**:  
The main user interface class for the hotel management system. This class offers a console-based menu where users can interact with different parts of the system, such as managing guests, rooms, bookings, and payments.

**Key Methods**:

* **mainMenu**: Displays the main menu with options for managing guests, rooms, bookings, and payments.
* **start**: Starts the application and manages the flow of the application by accepting user input.
* **guestMenu**: Allows the user to manage guest information.
* **roomMenu**: Provides options to manage room information, such as adding rooms and viewing room availability.
* **bookingMenu**: Manages booking-related operations, including viewing all bookings, adding new bookings, and viewing total income.
* **paymentMenu**: Provides payment-related options, including adding payments and viewing all payments.
* **addGuest**, **addRoom**, **addBooking**, **addPayment**: Methods to add new entities (guests, rooms, bookings, and payments) to the system.
* **getTotalIncome**: Displays the total income from bookings on a specific date.
* **deleteAllGuests**, **deleteAllRooms**, **deleteAllBookings**, **deleteAllPayments**: Methods that allow the user to delete all records of guests, rooms, bookings, and payments, respectively, with a confirmation step.

**3. Main Class**

**Purpose**:  
This is the entry point for the application. It initializes the database connection and creates the necessary controller and repository objects for handling the various system operations. It then starts the application, allowing the user to interact with the system.

**Key Actions**:

* Initializes a connection to the PostgreSQL database using the PostgresDB class.
* Creates instances of repositories (e.g., GuestRepository, RoomRepository, BookingRepository, PaymentRepository) that handle the data layer operations.
* Initializes controller instances (GuestController, RoomController, BookingController, PaymentController) that interact with the repositories.
* Starts the HotelApplication, allowing the user to access different features such as managing guests, rooms, bookings, and payments.

**4. PostgresDB Class (and IDB Interface)**

**Purpose**:  
The PostgresDB class is an implementation of the IDB interface, which provides the necessary database connections to interact with PostgreSQL. It encapsulates the connection logic, ensuring that database operations can be performed safely and efficiently.

**Key Methods**:

* **getConnection**: Establishes a connection to the PostgreSQL database using the provided connection string, username, and password.
* **close**: Closes the database connection when it's no longer needed.

**5. Controllers (GuestController, RoomController, BookingController, PaymentController)**

**Purpose**:  
The controller classes are responsible for handling user requests and performing business logic related to guests, rooms, bookings, and payments. They act as intermediaries between the view (user interface) and the model (repositories).

**Key Responsibilities**:

* **GuestController**: Manages operations related to guests, such as retrieving guest information and creating new guests.
* **RoomController**: Handles room-related operations, including adding rooms, viewing all rooms, and checking room availability.
* **BookingController**: Manages booking-related functionalities like creating new bookings, calculating total income, and retrieving all bookings.
* **PaymentController**: Manages payment-related actions such as adding payments and deleting payments.

**6. Repositories (GuestRepository, RoomRepository, BookingRepository, PaymentRepository)**

**Purpose**:  
Repository classes handle the data layer of the application by interacting directly with the database. They implement the respective interfaces and provide methods for CRUD operations related to guests, rooms, bookings, and payments.

**Key Responsibilities**:

* **GuestRepository**: Interacts with the database to manage guest information, such as retrieving all guests and adding new ones.
* **RoomRepository**: Manages room data in the database, including creating, updating, retrieving, and deleting room records.
* **BookingRepository**: Responsible for managing booking-related data, including adding bookings and calculating total income for a specific date.
* **PaymentRepository**: Handles payment data, including adding payments and deleting payment records.